# From Analysing Operating System Vulnerabilities to Designing Multiversion Intrusion-Tolerance Architectures

Anatoliy Gorbenko, Alexander Romanovsky, Olga Tarasyuk, Oleksandr Biloborodov

***Abstract*— This paper analyses security problems of modern computer systems caused by vulnerabilities in their operating systems. Our scrutiny of widely used enterprise operating systems focuses on their vulnerabilities by examining the statistical data available on how vulnerabilities in these systems are disclosed and eliminated, and by assessing their criticality. This is done by using statistics from both the National Vulnerabilities database (NVD) and the Common Vulnerabilities and Exposures system (CVE). The specific technical areas the paper covers are the quantitative assessment of forever-day vulnerabilities, estimation of days-of-risk, the analysis of the vulnerabilities severity and their distributions by attack vector and impact on security properties. In addition, the study aims to explore those vulnerabilities that have been found across a diverse range of operating systems. This leads us to analysing how different intrusion-tolerance architectures deploying the operating system diversity impact availability, integrity and confidentiality.**

***Index Terms*—security, vulnerability, operating systems, vulnerability databases, days-of-risk, forever-day vulnerabilities, vulnerability statistics, diversity, intrusion tolerance**

## I. Introduction

IT is of vital significance for system users and developers alike that information and communication systems are secure. There have been a numbers of occasions recently, such as those involving Hollywood Presbyterian Medical Center [1], San Francisco Municipal Transportation Agency [2] or British NHS [3], which have illustrated how exposed modern society is to attacks. The costs of such global cyberattacks as Petya or WannaCry could amount to millions of dollars, harm to our health and survival and damage to critical infrastructures [4]. It is because our communication equipment, computer systems and other smart devices suffer from software vulnerabilities that cyberattacks, malware intrusions and virus infections have been successful.

In general terms, a vulnerability is understood as a weakness that makes it possible for an intruder to damage the information assurance in a system. It has been defined as a software fault that a hacker can employ to access to a network or system (MITRE Corporation, [5]). There are various ways in which vulnerability can be exploited. Attackers can get commands executed in the normal way, or overcome restrictions in order to gain forbidden access to data, or trigger denial of service and system service termination. The primary source of software vulnerabilities is weaknesses and faults in software design and implementation. Of the 372 updates issued by Microsoft in 2017 for their operating systems, 228 were security updates for eradicating software vulnerabilities [6]. Of these, 137 were classified as critical.

Both operating systems (OSes) and application software can contain vulnerabilities, yet it is without doubt security flaws in OSes that are most critical since if they are exploited by attackers, all services and processes executed by the OS can be compromised and illicit access gained to any data that is stored on the exposed machine. Moreover, the threats they pose to system dependability and security are distinct from failures, faults and errors that have been the traditional focus of the dependability community's efforts.

For instance, in the beginning of May 2017 a global cyber attack using ransomware called Wanna Decryptor (also known as WanaCrypt0r 2.0, WannaCry or WCry) infected more than 300000 computers in 150 countries, hitting international shipper FedEx, large telecommunications companies in Spain, Portugal and Argentina, German railway operator Deutsche Bahn, etc. In Britain, the National Health Service (still widely using Windows XP OS in their IT systems) was the worst hit. Many UK hospitals and surgeries were forced to turn away patients and cancel appointments after their IT systems were infected with the ransomware. The attack was initiated through exploiting SMB vulnerability MS17-010 in Microsoft Windows family of operating system.

This paper builds on a number of studies which examine a range of OS security and vulnerability issues [7, 8, 9, 10]. Our investigation of some novel aspects of security could yield insights that would be significant for not only system administrators, security engineers and OS vendors but also

Manuscript received January 31, 2018; revised MM DD, 2018 and MM DD, 2018; accepted MM DD, 2018. Associate Editor: F. Sname. (Corresponding author: Anatoliy Gorbenko). This work was supported in by the EPSRC/UK STRATA platform grant. Anatoliy Gorbenko and Olga Tarasyuk are partially supported by the TEMPUS ALIOT grant.

A. Gorbenko is with Leeds Beckett University, Leeds, UK (e-mail: A.Gorbenko@leedsbeckett.ac.uk).

A. Romanovsky is with Newcastle University, Newcastle-upon-Tyne, UK (e-mail: Alexander.Romanovsky@ncl.ac.uk).

O. Tarasyuk is with the National Aerospace University, Kharkiv, Ukraine (e-mail: O.Tarasyuk@csn.khai.edu).

O. Biloborodov is with the Plarium Ukraine LLC, Kharkiv, Ukraine (e-mail: alexander.bright@hotmail.com)

ordinary users. It focuses on:

1) comparing, by using quantitative analysis and statistics, the vulnerabilities in a number of OSes that have been identified and resolved;

2) assessing the most significant vulnerability metrics including days-of-risk [11], numbers of forever-day [12] vulnerabilities and their severity for each operating system.

This paper expends our early work in [13] in a number of ways. First of all, we investigate addition important aspects, such as vulnerability distributions by attack vectors and their impact on different security properties (availability, confidentiality and integrity); a correlation between vulnerability severity and vendor's rapidity to fix them; analysing which types of vulnerabilities are the most numerous and severe. In addition, we use the reported statistics to examine intrusion-tolerance architectures aimed at improving system security using diversity of operating system and study how diversity can impact surface of attacks targeting different security attributes via common vulnerabilities. Lastly, we update our early study by adding 2017's vulnerability statistics.

There have been many works, e.g. [14, 15, 16], studying software diversity as a means for tolerating software faults since the 70s when the concept of N-version programming [17] and Recovery Blocks [18] were introduced. Software diversity has been successfully applied in various application domains, including railway, aerospace and nuclear power station control to improve system reliability.

One of the most challenging parts of the work on applying diversity in practice is the justification of the effectiveness of proposed solutions due to the lack of empirical data. The use of software diversity for security and intrusion-tolerance was proposed in earlier studies reported in [19, 20, 21, 22], which clearly showed the needs for demonstrating the applicability of the proposed architectural solutions and for evaluating their advantages to drive their design.

Our paper continues a series of works quantitatively studying common vulnerabilities of intrusion-tolerance systems employing OS diversity, e.g. [23, 24]. In spite of some similarities between our work and these studies, there are substantial differences. Firstly, Garcia et al. do not consider vulnerability statistics in dynamics taking into account a lag between the times when a vulnerability is disclosed and when OS vendor issue a patch to fix it. Secondly, in our work we analyse additional vulnerability metrics related to different OSes: average days-of-risk, average number of forever-day vulnerabilities, their types and severity. In addition, we examine how OS diversity and common vulnerabilities influence the attack surface and impact various security attributes of the specific intrusion-tolerance architecture. The reported statistics will help system administrators and users to make a justified decision when facing a challenge of choosing the most secure and the least vulnerable operating system and their combinations.

The rest of the paper is organized as follows. In the next section we briefly describe vulnerability databases and studied OSes, discuss the most important vulnerability measures (days-of-risk, forever-day vulnerabilities and their CVSS severity), present vulnerabilities discovery and patching statistics, and outline the most severe types of vulnerabilities as well as the vulnerabilities, discovered in more than one OS. Section III examines diverse intrusion-tolerance architectures and discusses how diversity of OSes affects various security properties: availability, integrity and confidentiality.

The final part, Section IV, sums up several practical conclusions to be drawn from our study.

## II. Background and Research Methodology

The main focus of our paper is to consider dynamical aspects of vulnerability life cycle. In particular, we study how often new vulnerabilities are discovered, how quick vendors issue patches, fixing vulnerabilities, and how many of yet unfixed vulnerabilities exist in a particular operating system at once. With this purpose, our research methodology relies on:

--collecting vulnerability statistics from different datasets and merging them in a single SQL-like database;

--considering the vulnerability life cycle and disclosure policies which are used by different vulnerability datasets;

--using the Common Platform Enumerations (CPE, https://cpe.mitre.org/dictionary/) corresponding to the studied OSes to filter vulnerability statistics from the database.

### A. Vulnerability Databases and Datasets

There are a wide range of actors that are investing plenty of effort into discovery and elimination of vulnerabilities, including software vendors, international governmental and non-governmental organizations, businesses and individuals.

Many of them make their vulnerability datasets publicly available. Among the most reputable of these are:

--CVE, the Common Vulnerabilities and Exposures system, is a list of established vulnerabilities maintained by MITRE Inc. (cve.mitre.org). Each vulnerability is assigned a unique identifier, CVE-ID, that other vulnerability databases use to synchronize their data with CVE and thus make data exchange between security databases and products possible. Over 18,000 of these identifiers were assigned by MITRE in 2017 alone. The vulnerability description provided in CVE is, however, rather basic and does not include such significant details as a comprehensive list of vulnerable products, vulnerability type and severity.

-- NVD, the National Vulnerability Database maintained by the U.S. National Institute of Standards and Technology (web.nvd.nist.gov), builds on and is synchronized with CVE. Unlike CVE, it categorises vulnerabilities by type and severity, provides a specific list of vulnerable software products and additional meta-data following the Common Platform Enumeration Dictionary (CPE), the Common Weakness Enumeration Specification (CWE) and the Common Vulnerability Scoring System (CVSS).

--VNDB, the Vulnerability Notes Database maintained by CERT (www.kb.cert.org/vuls/).

--VulnDB, a vulnerability database offered as a commercial product by the Risk Based Security company (www.riskbasedsecurity.com/vulndb/), can track weaknesses in third-party libraries.

--SecurityTracker, a vulnerability dataset available to buy

at securitytracker.com.

Another common way to inform customers about vulnerabilities in software products is vendors publishing security bulletins (e.g. https://technet.microsoft.com/en-us/security/bulletins.aspx). However, the previously widely used OSVDB (Open Source Vulnerability Database) and FVDB (Frei's Vulnerability Database) are not accessible any more.

NVD and CVE, the most comprehensive and reliable databases, make vulnerability data available by providing a simple search interface on their websites or daily updated XML data feeds. It would be difficult, however, to directly use their datasets for complex analytics since SQL queries are not supported.

### B. Vulnerability Life Cycle and CVE/NVD Disclosure Policies

There have been several studies focusing on the software vulnerability life cycle [25, 26, 11]. In one study [27] its most important milestones were defined in order to put forward its formal model. The common consensus among security analysts and researchers single out 5 major events which make up a typical vulnerability life cycle: (i) a vulnerability is created; (ii) it is discovered; (iii) it is disclosed; (iv) a patch is created; (v) the patch is installed.

The risks of system exposure for time intervals between these events tend to differ. Thus, there is a time of a higher security risk from the moment of vulnerability discovery or disclosure till the moment when a patch is installed to resolve it, referred to as *days-of-risk* [11]. The terms *black*, *grey* and *white risk* are used to refer to varying levels of exposure risk and of public awareness of the dangers involved (see Fig. 1). This paper deals with *grey (post-disclosure) risk* associated with the interval between the vulnerability being disclosed and the patch to fix it being provided.

The paper takes the date when a vulnerability is assigned a CVE-ID in CVE as vulnerability disclosure time. This is because CVE-IDs are unique identifiers, whereas most other security bulletins and vulnerability databases are seen as secondary since their records are synchronized with them.

While it is sometimes possible to derive the time when a patch is produced from vendors' security bulletins, more commonly it is necessary to search vendors' web sites manually in order to extract the relevant information, since typically there are no reporting mechanisms or xml-based data feeds that would allow automatic search and processing.

It has been reported [28] that, for about 75% of vulnerability descriptions, the median time from the moment when they appear in vendor security bulletins till the time when NVD makes them available is seven days. This suggests that NIST allows time for a patch to be produced to fix the vulnerability before publishing the detailed information in NVD, implementing what has been called a *responsible disclosure model* [29]. In addition, the median announcement gap varies depending on the vendor: it is 2 days for Microsoft, 5 days for Oracle and Apple, 10 days for Linus and 12 days for Novell.

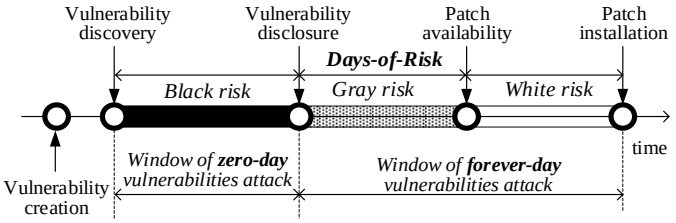


Fig. 1. Vulnerability lifecycle.

### C. Operating Systems Under Study

This study examines the vulnerabilities of six widely used enterprise operating systems (see Table I). Our reasons for choosing these particular OSes and their versions included their popularity, the fact that they include both proprietary and open-source types, belong to different families (Windows, Unix/Linux, MacOS), and are sold by different vendors for a range of application domains. This prompted us to consider a series of studies (e.g. [30, 31, 32]) focusing on the OS market share of web servers, where Linux-based OSes predominate, and of on-premises server, where various versions of Microsoft Windows are most common.

Our aim was to examine vulnerability data over a significant period in order to identify major trends. We also wanted to ensure that our conclusions are based on comprehensive datasets (in NVD and CVE, there is not enough information on the most recent OS versions for statistical analysis). For these reasons, the choice of OS versions was made (see Table I) so as to focus our scrutiny on the six years between the late 2011 and the late 2017, analysing a total of over 2,500 vulnerabilities. Even though the OS versions selected have already been replaced by more recent ones, our research demonstrates that new vulnerabilities are still being discovered in the older OS versions. Furthermore, most of these new vulnerabilities can also be found in the latest versions of OSes.

To precisely identify vulnerabilities discovered in a particular operating system we use the Common Platform Enumeration Dictionary (CPE) [33]. The CPE dictionary, maintained by NIST and used by NVD, offers is a structured hierarchical naming scheme and a generic syntax for identifying computer systems, software, and packages. Each vulnerability record stored in NVD has a list of CPE references which allows exact identification of all vulnerable products.

Each CPE reference uses the following general syntax: *cpe:/type-of-product {o – operating system | a – application software | h – hardware/firmware}:manufacturer:product-name :release:version:subversion(s):platform{x64|x86}.*

TABLE I.
OPERATING SYSTEMS UNDER INVESTIGATION

| Operating system | Release date | Linux kernel version | No of CPE references |
|---|---|---|---|
| Ubuntu Server 12.04 | 26.04.2012 | 3.2.x | 82 |
| Red Hat Enterprise Linux 6 | 10.11.2010 | 2.6.32.x | 87 |
| Novell Linux SUSE Enterprise Server 11 SP2 | 27.02.2012 | 3.0.13 | 58 |
| Microsoft Windows Server 2012 R2 | 18.10.2012 | - | 12 |
| Apple MacOS Server 10.8 | 25.06.2012 | - | 7 |
| Oracle/Sun Solaris 11 | 09.11.2011 | - | 9 |

In practice, several CPE references can match the same product (e.g. some of CPEs can refer to the whole family of OSes while others can identify the certain OS, version or release). For example, the list of CPE references corresponding to Microsoft Windows Server 2012 R2 consists of 12 entries including:

--cpe:/o:microsoft:windows:::~~~~x64~
--cpe:/o:microsoft:windows:::~~~~x86~
--cpe:/o:microsoft:all_windows:abstract_cpe
--cpe:/o:microsoft:all_windows
--cpe:/o:microsoft:windows_server_2012:r2::~~~x64~~
--cpe:/o:microsoft:windows_server_2012:-, etc.

Lists of CPEs for Ubuntu, Red Hat and Novell OSes should also be supplemented with CPE entries corresponding to Linux kernels used by each of these OSes. Being a part of an operating system a Linux kernel, nevertheless, is considered by NVD as a separate software product having own vulnerabilities. For example, the list of CPEs assigned to Linux kernel 3.2.x used by Ubuntu Server 12.04 includes:

--cpe:/o:linux:linux_kernel
--cpe:/o:linux:linux_kernel:-
--cpe:/o:linux:linux_kernel:3.2
--cpe:/o:linux:linux_kernel:3.2::~~~~x86~
--cpe:/o:linux:linux_kernel:3.2:rc2
--cpe:/o:linux:linux_kernel:3.2.1
--cpe:/o:linux:linux_kernel:3.2.1::~~~~x86~, etc.

In our study, the number of vulnerabilities in Linux kernels represents on average 40% of the total number of vulnerabilities disclosed during 2012-2017 in Ubuntu, Red Hat and Novell.

### D. *Research methodology*

Our research methodology is presented in Fig. 2. It consists of seven steps including:

***Step 1:*** first, we designed and created a MySQL database to aggregate information from the CVE and NVD databases.

***Step 2:*** we developed a software tool which merges together XML data files provided by CVE and NVD, and inserted the joint data set into the MySQL database. The tool consistently updates our MySQL vulnerability database by:

--downloading XML data feeds from CVE and inserting all new vulnerabilities into the MySQL database, using CVE-ID as a primary key and the CVE date as a vulnerability disclosure time (Step 2.1);

--downloading XML data feed from NVD and, if necessary, update vulnerability records existed in the MySQL database by CVE-ID (Step 2.2). In particular, if NVD reports a new vulnerability we set the NVD date as the time when a vulnerability is fixed by a vendor and add CVSS, CWE and CPE information from NVD in addition to that previously imported from CVE.

Thus, our MySQL database stores both dates associated with the same vulnerability: (i) when a vulnerability is first announced by CVE and (ii) when its description appears in NVD. This allows us to estimate the period of *grey risk*. We did not exclude NVD announcement gaps, discussed at the end of Section II.B, during which vulnerability descriptions propagate from vendor's security bulletins to the NVD database. This can result in a slightly pessimistic estimate, which, nevertheless, seems to be more secure than their underestimate. Because CVE and NVD are updated daily, the tool performs steps 2.1 and 2.2 every day. By now, our MySQL database includes more than 100000 vulnerability records.

***Step 3:*** at this step we selected six popular server operating systems which vulnerabilities we wanted to examine.

***Step 4:*** we used the CPE Dictionary to create 6 lists of CPE references corresponding to each operating system. Table I reports how many of CPE entries have been associated with each operating system (the lists themselves can be downloaded from GoogleDrive[1]).

***Step 5:*** the CPE lists created at the previous step were used to query the MySQL database and select a subset of vulnerabilities belonging to certain OSes.

***Step 6:*** at this stage we run a series of sub-requests to collect various vulnerability statistics reported in Section III.

***Step 7:*** at the final step we studied common OSes vulnerabilities (by analysing overlaps of the lists of CPE entries assigned to each vulnerability) and investigated how diversity of OSes affects system availability, integrity and consistency.

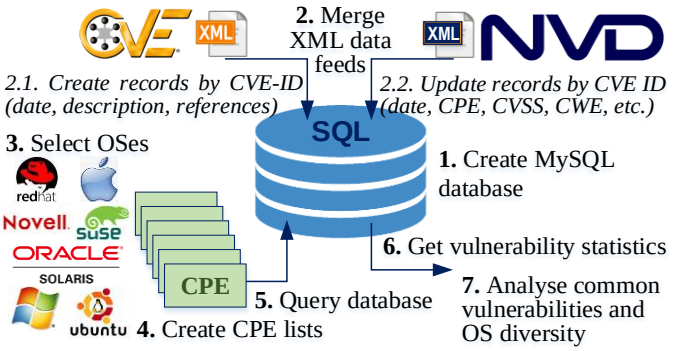


Fig. 2. Research methodology.

The accuracy of the results reported in our work fully depends on the accuracy of the data, reported by CVE and NVD. As we mentioned earlier, CVE and NVD are highly reputable vulnerability databases, widely used by many researchers and security analysts, that also provide data feeds for the third-party security tools (e.g. vulnerability scanners). Moreover, we assume that MITRE Inc. and NIST, operating CVE and NVD, spend comparatively equal efforts on examining vulnerability of different software products and provide trusted information that can be used as an indicator of software security/quality.

## III. OSes Vulnerability Study

### A. *Vulnerability Discovery, and Patching Statistics and Dynamics*

In this section we summarize the statistics of vulnerabilities discovered and fixed in different OSes since the 1st of January 2012 and until the 31st of December 2017 (see Table II). In the table we use the following short names for the operating systems under investigation:

[1] https://drive.google.com/open?id=1rToATBng3D4vGL7P7bnoxSywsKe0rDdW

--Ubuntu – Ubuntu Server 12.04.
--Red Hat – Red Hat Enterprise Linux 6.
--Novell – Novell Linux Enterprise Server 11 SP2.
--Windows – Microsoft Windows Server 2012 R2.
--MacOS – Apple MacOS Server 10.8.
-- Solaris – Oracle Solaris 11.

Red Hat Enterprise Linux 6 and Oracle Solaris 11 had been released before the observed period (see Table II). Other operating systems (Ubuntu Server 12.04, Novell Linux Enterprise server 11 SP2, Microsoft Windows Server 2012 R2 and Apple Macintosh Server 10.8) were released in the beginning of 2012. It is worth mentioning that on the date of the official release some of those operating systems already had vulnerabilities that earlier had been discovered in previous OS versions. In particular, Ubuntu Server 12.04 inherited 15 of such vulnerabilities, Red Hat Enterprise Linux 6 – 46, Novell Linux Enterprise server 11 SP2 – 26 and Oracle Solaris 11 – 13 vulnerabilities. Such vulnerabilities are reported as 'Inherited' in the Table II.

During 2012-2017 the largest number of vulnerabilities (1034) was disclosed in Ubuntu, the least number (205) – in MacOS. The Red Hat and Novell operating systems occupy a middle position having 560 and 415 vulnerabilities, respectively. The greatest number of vulnerabilities in 2007 (190) were discovered in Windows. Cumulative graphs of vulnerabilities disclosed via the CVE and NVD databases during 2012-2017 in studied OSes are depicted in Fig. 3. One can notice that reporting mechanisms are quite different for different OSes. In particular, curves depicting the numbers of vulnerabilities discovered in Linux-based OSes are less discrete having considerably more "small steps". This means that vulnerabilities are reported quite often by small portions or even individually. At the same time, vulnerabilities in Windows, Solaris and MacOS are usually batch-reported only few times a year.

TABLE II.
VULNERABILITY DISCLOSURE STATISTICS

| Year | Vulnerabilities | Ubuntu | Windows | Red Hat | Novell | MacOS | Solaris |
|---|---|---|---|---|---|---|---|
| Inherited | | 15 | 0 | 46 | 26 | 0 | 13 |
| 2012 | Disclosed | 64 | 10 | 31 | 32 | 2 | 47 |
| | Fixed | 32 | 5 | 40 | 36 | 2 | 47 |
| | Avg.Sev. | 5.21 | 8.31 | 5.07 | 5.13 | 3.20 | 4.37 |
| 2013 | Disclosed | 196 | 59 | 71 | 124 | 60 | 31 |
| | Fixed | 202 | 51 | 86 | 127 | 59 | 32 |
| | Avg.Sev. | 5.04 | 7.12 | 5.09 | 4.94 | 4.92 | 4.72 |
| 2014 | Disclosed | 188 | 64 | 72 | 129 | 40 | 37 |
| | Fixed | 197 | 38 | 58 | 107 | 40 | 35 |
| | Avg.Sev. | 5.55 | 6.71 | 6.79 | 5.83 | 7.13 | 5.08 |
| 2015 | Disclosed | 251 | 178 | 162 | 52 | 14 | 74 |
| | Fixed | 223 | 156 | 146 | 80 | 15 | 71 |
| | Avg.Sev. | 6.06 | 6.66 | 5.75 | 6.67 | 6.53 | 5.12 |
| 2016 | Disclosed | 214 | 204 | 125 | 23 | 48 | 4 |
| | Fixed | 238 | 156 | 16 | 34 | 48 | 17 |
| | Avg.Sev. | 5.50 | 5.93 | 6.87 | 7.17 | 6.78 | 6.60 |
| 2017 | Disclosed | 106 | 190 | 53 | 29 | 41 | 5 |
| | Fixed | 79 | 234 | 59 | 25 | 24 | 9 |
| | Avg.Sev. | 6.25 | 4.50 | 6.05 | 6.19 | 4.04 | 5.18 |
| Total | Disclosed | 1034 | 705 | 560 | 415 | 205 | 211 |
| | Fixed | 971 | 640 | 405 | 409 | 188 | 211 |
| | Avg.Sev. | 5.60 | 6.54 | 5.94 | 5.99 | 5.43 | 5.18 |

### B. Days-of-Grey-Risk Statistics

The number of disclosed vulnerabilities is often used as the major indicator of software insecurity. However, taking into account how fast software vendors react on vulnerabilities discovered in their products is equally important. To compare efforts that different vendors make to solve security issues and to deliver security updates fixing vulnerabilities we use the *Days-of-Risk* measure.

*Days-of-risk* [11] defines a period of time after a vulnerability is discovered/disclosed and until it is eliminated from a system after patch installation. It is also known as '*window of-vulnerability*' or '*days-of-recess*'. In this study we do not take into account possible delays between the times when a vendor issues the patch and until a user or a system administrator actually installs it.

Besides, in many cases it is impossible to identify when exactly a vulnerability was discovered. In the paper we investigate, so called, *gray risk* or *post-disclosure risk* which defines the interval between vulnerability disclosure time and the date when the patch fixing vulnerability becomes available [13, 11]. In accordance with our research methodology, discussed in Section II.D we estimate *days-of-gray-risk* (DoGR) for a particular vulnerability as the period of time between a vulnerability is initially reported in CVE and its description appears in NVD.

Table III shows how the average days-of-risk have been changing during 2012-2017 for different operating systems. It also includes data reported by other researchers in [7, 10, 9, 34] for earlier versions of the studied OSes. For instance, according to [10] in 1999 Microsoft spent an average 16 days from vulnerability disclosure to issuing a patch. Red Hat spent only 11 days to fix vulnerabilities while Sun proved itself to be very slow solving security problems in 90 days on average.

In 2006, as reported in [7, 34], the days-of-gray-risk parameter for Microsoft Windows series of operating systems (Windows 2000 Professional and Server, Windows XP, Windows Server 2003) was estimated at 29 in average. At the same time, it took Red Hat 107 days to deliver security updates for its Enterprise Linux 2.1, 3.0 and 4.0 while Sun spent 168 days to do the same for any Solaris version patched in 2006. In addition, it was estimated that Apple Mac OS X and Novell SUSE Linux Enterprise Server and Desktop (versions 8–10) had 46 and 74 days-of-gray-risk respectively.

TABLE III.
AVERAGE DAYS-OF-GRAY-RISK STATISTICS

| Year | Ubuntu | Windows | Red Hat | Novell | MacOS | Solaris |
|---|---|---|---|---|---|---|
| 1999* | - | 16 | 11 | - | - | 90 |
| 2005** | - | 24 | 90 | 68 | 55 | 159 |
| 2006** | - | 29 | 107 | 74 | 46 | 168 |
| 2012 | 144 | 132 | 243 | 109 | 94 | 89 |
| 2013 | 109 | 131 | 119 | 99 | 113 | 81 |
| 2014 | 62 | 100 | 108 | 68 | 107 | 69 |
| 2015 | 79 | 126 | 101 | 133 | 83 | 58 |
| 2016 | 105 | 183 | 130 | 144 | 138 | 210 |
| 2017 | 34 | 89 | 80 | 36 | 225 | 49 |

*taken from [10]; **taken from [7], [9] and [34].

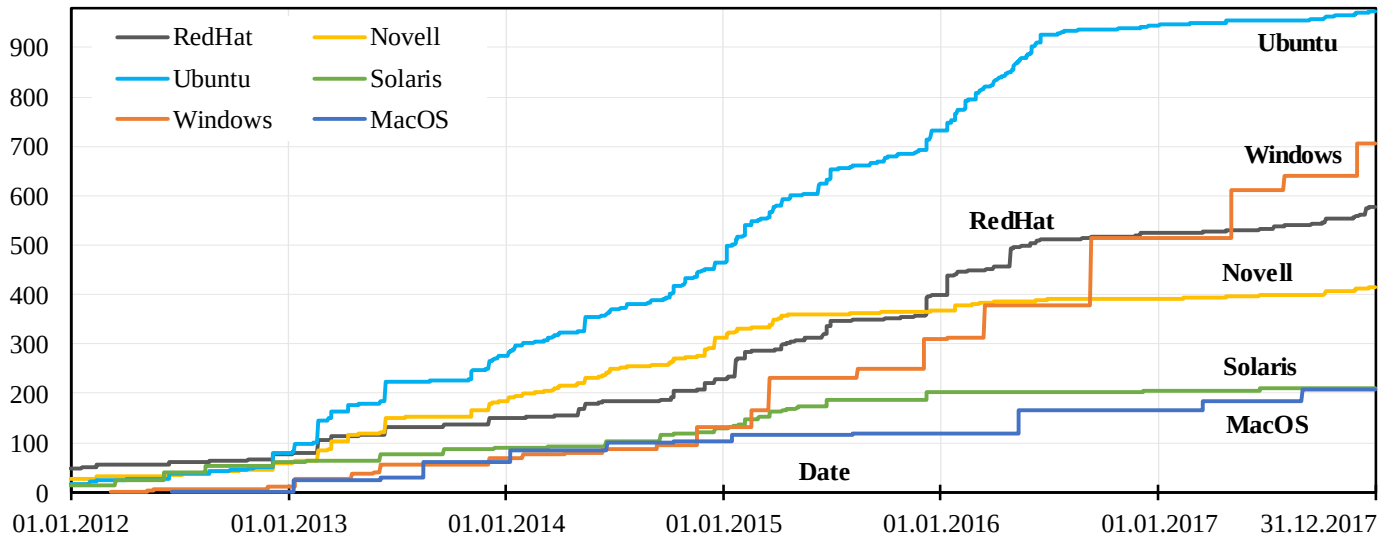


Fig. 3. Cumulative number of disclosed vulnerabilities.

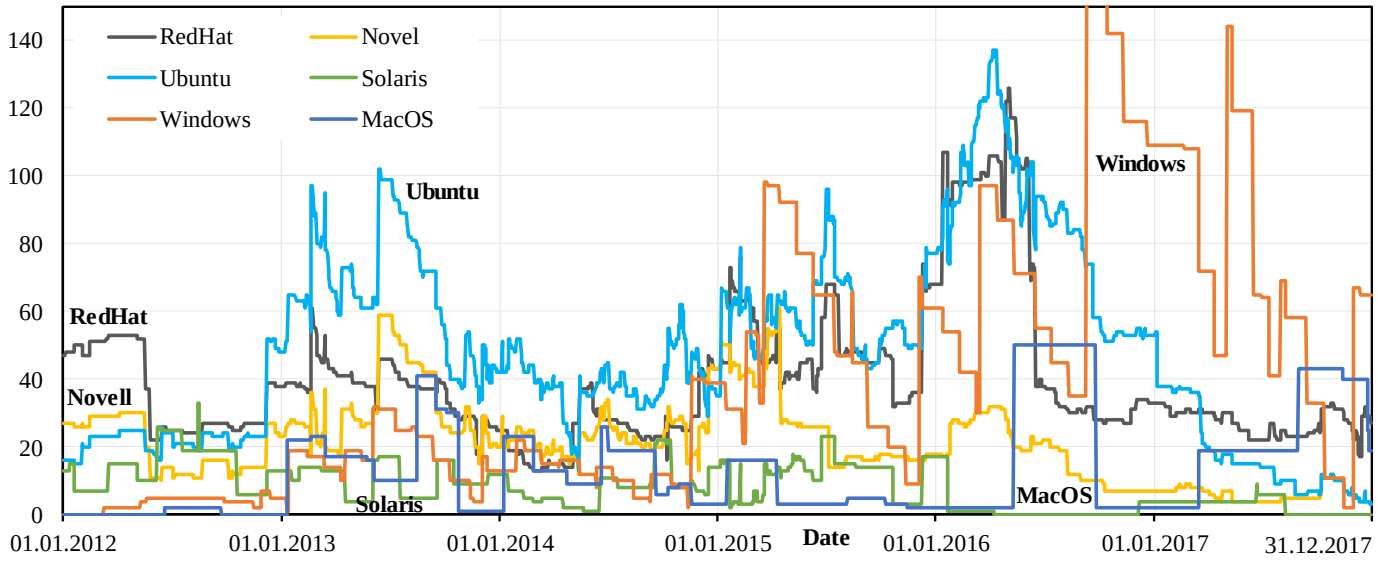


Fig. 4. Forever-day vulnerabilities.

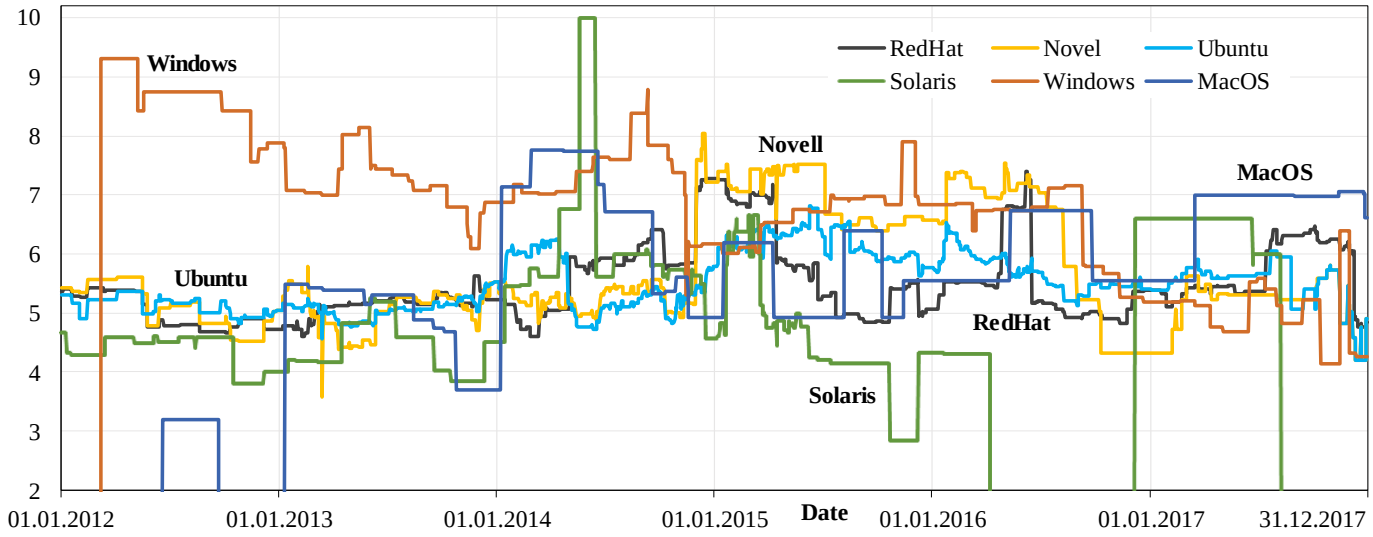


Fig. 5. Average severity of forever-day vulnerabilities.

Table III shows that since 2012 (excepting 2017) there has been a general tendency towards shortening the period of grey risk. However, during the last two years, the average days-of-grey-risk for different operating systems varies significantly between 34 and 225 days. Unfortunately, it still means that after a vulnerability public disclosure users of affected operating system remain vulnerable and unprotected against potential hacker attacks during months, and the OS vendors are aware of this.

Our work clearly shows that the conclusion by Jeff Jones expressed in a series of his earlier blog posts [7, 11, 35] that Windows is the platform exposing users to risks for the shortest period of time as compared to other OSes is no longer correct.

At the same time, we can see that since Oracle took ownership of Solaris OS in 2009 the Solaris OS has demonstrated the steady reduction of days-of-gray-risks. This let us to conclude that Oracle has been reacting on new vulnerabilities much faster than Sun did.

### C. Forever-Day Vulnerability Statistics

The authors of [12] coin a new term '*forever-day vulnerability*' defining a publicly disclosed vulnerability that has not been patched yet and can be hacked any time during system operation. It is in contrast to '*zero-day vulnerabilities*' [27] which are publically undisclosed vulnerabilities that some hackers have already discovered and can exploit.

Using both, the date of vulnerability disclosure and the date when the OS vendor issues a patch to fix it we can plot graphs of *forever-day vulnerabilities* showing how many of known (already disclosed publicly) but yet unfixed vulnerabilities existed every day during 2012-2017 in a particular operating system (see Fig. 4). Any operating system running with forever-day vulnerabilities is always vulnerable unless the software vendor issues a patch and a system administrator installs it.

Usually, software vulnerabilities are disclosed much faster than vendors manage to fix them. This is why a particular operating system can contain up to several dozens of forever-day vulnerabilities at a time. Any of these vulnerabilities could be potentially exploited by hackers to attack the system. Fig. 4 shows that some operating systems have only few days (if any) of vulnerability free operation per year.

For instance (see Table IV), during 2012–2017 OS Ubuntu, Windows, Red Hat and Novell did not have known vulnerability free days at all. MacOS had only 111 of such days. It is our hope that OS users and administrators understood and accepted the potential risk of running these systems. In addition, Table IV presents a detailed statistics of forever-day vulnerabilities for each operating system during 2012-2017. On average, Ubuntu OS had 48 of such vulnerabilities every day. OS Windows and Red Hat had 40 forever-day vulnerabilities on average (twice as many as Novell). MacOS and Solaris had the least average number of forever-day vulnerabilities (13 and 8 respectively).

### D. Vulnerability Severity and CVSS-based Statistics

Quantitative evaluation of computer systems vulnerability is a question of great debates with many approaches proposed [36, 37, 38, 39]. It is clear that the more vulnerabilities exist in a system, the more that system is prone to hacker attacks.

TABLE IV.
FOREVER DAY VULNERABILITIES STATISTICS

| Year | Forever day vulnerabilities | Ubuntu | Windows | Red Hat | Novell | MacOS | Solaris |
|---|---|---|---|---|---|---|---|
| 2012 | Min | 15 | 2 | 22 | 10 | 0 | 6 |
| | Max | 52 | 7 | 53 | 30 | 2 | 33 |
| | Average | 24 | 4 | 36 | 20 | 1 | 14 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 102 | 0 |
| 2013 | Min | 33 | 1 | 18 | 15 | 0 | 4 |
| | Max | 102 | 32 | 61 | 59 | 41 | 17 |
| | Average | 66 | 18 | 37 | 32 | 17 | 10 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 9 | 0 |
| 2014 | Min | 13 | 2 | 13 | 12 | 1 | 1 |
| | Max | 62 | 41 | 47 | 44 | 26 | 22 |
| | Average | 39 | 16 | 25 | 23 | 13 | 8 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 0 | 0 |
| 2015 | Min | 35 | 9 | 32 | 14 | 2 | 2 |
| | Max | 96 | 98 | 76 | 61 | 16 | 23 |
| | Average | 59 | 51 | 49 | 27 | 6 | 12 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 0 | 0 |
| 2016 | Min | 51 | 30 | 27 | 7 | 2 | 0 |
| | Max | 137 | 171 | 126 | 32 | 50 | 17 |
| | Average | 85 | 83 | 62 | 18 | 20 | 1 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 0 | 242 |
| 2017 | Min | 3 | 2 | 17 | 3 | 2 | 0 |
| | Max | 54 | 144 | 33 | 12 | 43 | 9 |
| | Average | 17 | 68 | 27 | 6 | 23 | 3 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 0 | 146 |
| Total | Min | 3 | 0 | 13 | 3 | 0 | 0 |
| | Max | 137 | 171 | 126 | 61 | 50 | 33 |
| | Average | 48 | 40 | 40 | 21 | 13 | 8 |
| | Vuln. free days | 0 | 0 | 0 | 0 | 111 | 388 |

However, one should also account how quick a vendor fixes vulnerabilities, how critical vulnerabilities are, how they impact on security properties, etc.

Vulnerability *severity* is an important characteristic quantifying the impact of vulnerability on system security. NVD has adopted the Common Vulnerability Scoring System (CVSS) to assign severity scores to software vulnerabilities [40]. CVSS is composed of three metric groups: Base, Temporal and Environmental, each consisting of a set of metrics. The CVSS *Base score* represents the intrinsic and fundamental characteristics of a vulnerability independently of exploits and/or payloads. It is calculated using a group of qualitative metrics taking into account:

--attack vector (local, adjacent network, network);

--access complexity (high, medium or low);

--need for authentication (required or not; multiple or single);

--vulnerability impact on *confidentiality*, *integrity* and *availability* (none, partial or complete); some of vulnerabilities impact only one security attribute while others can lead to breaches in two or all three of them.

*Temporal* and *Environmental* scores are optional. They represent the characteristics of a vulnerability that can change over time (e.g. once the exploit code becomes available) and among user environments (e.g. whether a vulnerable system is exposed publically in the Internet or not).

In this section we consider only CVSS base scores provided by the NVD vulnerability database that are constant over time and user environments. Note that the CVSS vulnerability severity ranges from 0 to 10, with 10 being the most severe.

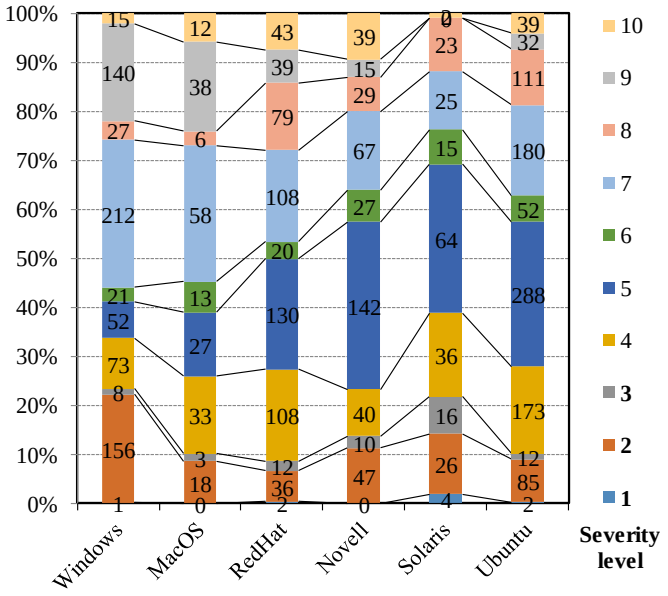


Fig. 6. Vulnerabilities distribution by CVSS severity scores.

The average CVSS vulnerability severity scores (Avg.Sev.) for different OSes are presented in Table II.

We could see, for example, that vulnerabilities in Oracle Solaris are the least critical with average severity equal to 5.18. The most severe vulnerabilities have been discovered in Microsoft Windows (the average severity is 6.54) and Novell (the average severity is 5.99).

Fig. 6 shows the percentage of vulnerabilities with different severity levels. Almost a quarter of vulnerabilities discovered in Microsoft Windows, MacOS and Red Hat are critical (e.g. their CVSS severity scores are in the range [8.0..10.0]). The lowest percentage of critical vulnerabilities (less than 12%) was observed in Solaris.

It is worth mentioning that *system vulnerability* is a dynamically changing characteristic. It changes every time when a new vulnerability is discovered in a system or when a patch fixing one of the previously discovered vulnerability is issued by a vendor and applied by a system administrator.

Thus, system vulnerability at a particular moment of time can be estimated as a product of the current number of *forever-day-vulnerabilities* (see Fig. 4) and their *average severity* (see Fig. 5).

As shown in Fig. 5 the severity of vulnerabilities disclosed in the Microsoft OS, having the highest value on average, nevertheless, tends to gradually decrease in time. In contrast, the severity of vulnerabilities in Linux- and Unix-based systems is gradually increasing. It is also worth noting that that there is no a strong correlation between the numbers of forever-day vulnerabilities observed in particular OS and their average severity.

Table V demonstrates vulnerabilities distribution among different CVSS criteria: attack vector, need for authentication and impact on security properties.

TABLE V.
CVSS-BASED VULNERABILITY STATISTICS

| Year | CVSS criteria | Vulnerabilities | Ubuntu | Windows | Red Hat | Novell | MacOS | Solaris |
|---|---|---|---|---|---|---|---|---|
| Inherited | Attack vector | Local | 6 | 0 | 28 | 19 | 0 | 6 |
| | | Adj. network | 2 | 0 | 4 | 1 | 0 | 0 |
| | | Network | 7 | 0 | 14 | 6 | 0 | 7 |
| | Auth. | Required | 1 | 0 | 1 | 1 | 0 | 2 |
| | | None | 14 | 0 | 45 | 25 | 0 | 11 |
| | Impact | Confidentiality | 7 | 0 | 15 | 12 | 0 | 7 |
| | | Integrity | 7 | 0 | 12 | 7 | 0 | 3 |
| | | Availability | 12 | 0 | 39 | 20 | 0 | 9 |
| 2012 | Attack vector | Local | 25 | 1 | 14 | 24 | 1 | 26 |
| | | Adj. network | 4 | 0 | 1 | 4 | 0 | 12 |
| | | Network | 35 | 9 | 16 | 4 | 1 | 9 |
| | Auth. | Required | 9 | 0 | 3 | 4 | 0 | 7 |
| | | None | 55 | 10 | 28 | 28 | 2 | 40 |
| | Impact | Confidentiality | 35 | 9 | 12 | 20 | 2 | 16 |
| | | Integrity | 25 | 9 | 17 | 13 | 0 | 18 |
| | | Availability | 45 | 8 | 24 | 25 | 0 | 39 |
| 2013 | Attack vector | Local | 109 | 31 | 34 | 88 | 27 | 21 |
| | | Adj. network | 11 | 1 | 3 | 11 | 1 | 0 |
| | | Network | 76 | 27 | 34 | 25 | 32 | 10 |
| | Auth. | Required | 24 | 2 | 7 | 12 | 3 | 5 |
| | | None | 172 | 57 | 64 | 112 | 57 | 26 |
| | Impact | Confidentiality | 120 | 47 | 48 | 72 | 39 | 7 |
| | | Integrity | 97 | 40 | 42 | 45 | 34 | 11 |
| | | Availability | 140 | 51 | 49 | 90 | 31 | 27 |
| 2014 | Attack vector | Local | 81 | 26 | 14 | 72 | 13 | 22 |
| | | Adj. network | 4 | 4 | 1 | 3 | 0 | 1 |
| | | Network | 103 | 34 | 57 | 54 | 27 | 14 |
| | Auth. | Required | 11 | 3 | 1 | 15 | 0 | 3 |
| | | None | 177 | 61 | 71 | 114 | 40 | 34 |
| | Impact | Confidentiality | 101 | 52 | 54 | 70 | 40 | 20 |
| | | Integrity | 79 | 44 | 50 | 54 | 35 | 14 |
| | | Availability | 144 | 44 | 60 | 106 | 34 | 30 |
| 2015 | Attack vector | Local | 32 | 105 | 14 | 12 | 9 | 31 |
| | | Adj. network | 0 | 1 | 1 | 1 | 0 | 0 |
| | | Network | 219 | 72 | 147 | 39 | 5 | 43 |
| | Auth. | Required | 41 | 4 | 48 | 4 | 0 | 5 |
| | | None | 210 | 174 | 114 | 48 | 14 | 69 |
| | Impact | Confidentiality | 147 | 158 | 95 | 37 | 12 | 31 |
| | | Integrity | 154 | 140 | 90 | 38 | 14 | 34 |
| | | Availability | 203 | 136 | 134 | 40 | 12 | 63 |
| 2016 | Attack vector | Local | 65 | 85 | 9 | 12 | 9 | 3 |
| | | Adj. network | 0 | 16 | 1 | 0 | 0 | 0 |
| | | Network | 149 | 103 | 115 | 11 | 39 | 1 |
| | Auth. | Required | 7 | 27 | 2 | 0 | 0 | 0 |
| | | None | 207 | 177 | 123 | 23 | 48 | 4 |
| | Impact | Confidentiality | 114 | 173 | 98 | 20 | 42 | 4 |
| | | Integrity | 103 | 123 | 94 | 19 | 26 | 4 |
| | | Availability | 178 | 134 | 99 | 22 | 35 | 4 |
| 2017 | Attack vector | Local | 21 | 132 | 27 | 19 | 3 | 3 |
| | | Adj. network | 2 | 0 | 2 | 2 | 1 | 0 |
| | | Network | 6 | 58 | 24 | 4 | 37 | 2 |
| | Auth. | Required | 0 | 8 | 1 | | 1 | 0 |
| | | None | 29 | 182 | 52 | 25 | 40 | 5 |
| | Impact | Confidentiality | 20 | 181 | 37 | 17 | 38 | 1 |
| | | Integrity | 21 | 81 | 37 | 18 | 27 | 3 |
| | | Availability | 27 | 87 | 46 | 23 | 29 | 3 |
| Total | Attack vector | Local | 339 | 380 | 140 | 246 | 62 | 112 |
| | | Adj. network | 23 | 22 | 13 | 22 | 2 | 13 |
| | | Network | 595 | 303 | 407 | 143 | 141 | 86 |
| | Auth. | Required | 93 | 44 | 63 | 36 | 4 | 22 |
| | | None | 864 | 661 | 497 | 375 | 201 | 189 |
| | Impact | Confidentiality | 544 | 620 | 359 | 248 | 173 | 86 |
| | | Integrity | 486 | 437 | 342 | 194 | 136 | 87 |
| | | Availability | 749 | 460 | 451 | 326 | 141 | 175 |

It shows, for instant, that 75% of vulnerabilities in Red Hat OS are network-exploitable; for Ubuntu, MacOS and Solaris a percentage of network vulnerabilities is over 50%; the fewest percentages of network exploitable vulnerabilities have been detected in Windows (46%) and Novell (40%). Ubuntu and Red Hat have the highest number of network-exploitable vulnerabilities (618 and 420 vulnerabilities correspondingly).

Practically it means, that it is undesirable to expose Ubuntu and Red Hat as web- or e-mail servers publicly available in the Internet because of a high chance to be hacked.

Another information of concern is a significant number of vulnerabilities (from 88% to 98% for different OSes) that do not require user authentication to be exploited. It means that most of hacker attacks would simply bypass built-in OS access control mechanisms making them useless.

Note here that the sum of vulnerabilities within the CVSS 'impact' metric group is higher than the total number of disclosed vulnerabilities presented in Table V. This is explained by the fact that most of vulnerabilities once exploited would allow an attacker to compromise at once all system security properties: confidentiality, integrity and availability.

### E. *Interdependency Between Vulnerability Severity and Days-of-Grey Risk*

Any software users would expect that vendors always try to fix the most severe vulnerabilities firstly. On the other hand, a rational vendor would take a risk-based view to decide which vulnerability to give high priority by taking into account the likelihood of exploit.

A vulnerability may be difficult to exploit (e.g. requires a very high competence or simply security controls commonly used make an exploit very difficult). Ignoring the likelihood of exploits from the vendor's point of view may be a recipe for wasting resources.

The CVSS *base score* can be considered as a good risk-based indicator as it integrates both *vulnerability impact metrics* (impact on integrity, confidentiality and availability) determining vulnerability severity and *exploitability metrics* (attack vector, access complexity and needs for authentication) which define the likelihood of exploits (https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator).

A set of box-and-whisker diagrams on Fig. 7 shows the numbers of days-of-grey-risk corresponding to vulnerabilities of different CVSS scores. They allow us to compare how quick OS vendors fix the least (CVSS severity score is in the range [1.0..3.0]) and the most (CVSS severity score is in the range [8.0..10.0]) critical vulnerabilities.

Unfortunately, it is shown that the days-of-risk metric does not actually depend on the CVSS vulnerability severity rating. The presented results disprove a widespread hypothesis that software vendors put more efforts into fixing the most critical vulnerabilities. To some extent it seems to be true for the Red Hat operating system. Windows spends approximately the same time to fix the most and the least severe vulnerabilities (127 vs 128 days on average). However, the developers of other OSes spend considerably more time on fixing critical vulnerabilities as compared to the least severe ones.

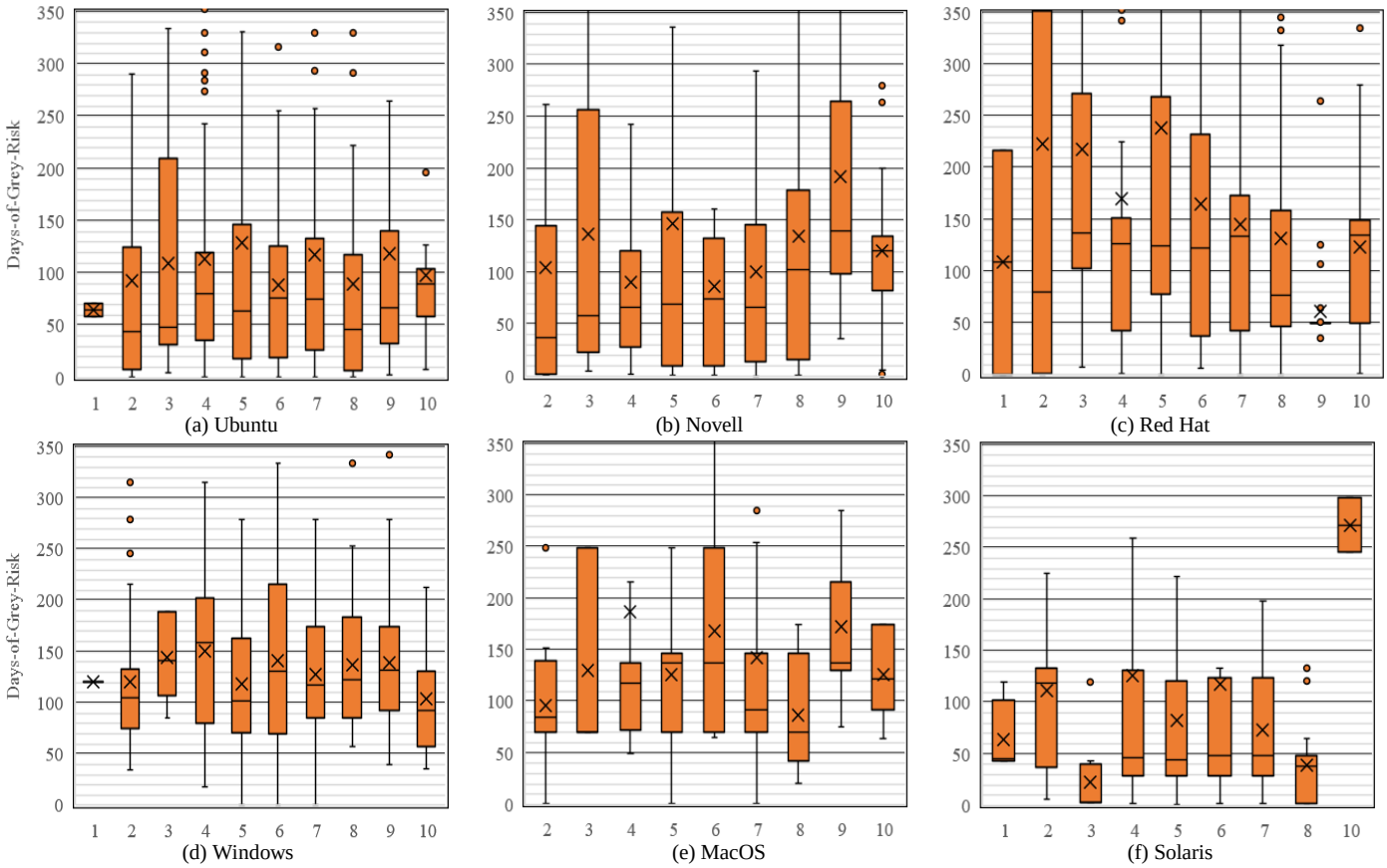


Fig. 7. Box-and-whisker diagrams showing a days-of-gray-risk statistics (Y-axes) for vulnerabilities of different CVSS severity scores (X-axes)

### *F. The Most Critical Types of OS Vulnerabilities*

NVD classifies all vulnerabilities using the Common Weakness Enumeration (CWE) scheme. CWE is a formal list of software weakness types proposed by MITRE Corporation (https://cwe.mitre.org/).

Our analysis demonstrates that the most numerous types of vulnerabilities for operating systems in general are:

CWE-119 (24%) – Improper restriction of operations within the bounds of a memory buffer caused by weaknesses of certain programming languages (often C and C++) that do not control bounds for the memory buffer that is being addressed. Vulnerabilities of the CWE-119 type usually cause arbitrary code execution, altering the intended control flow leading to accesses to protected information or system crash;

CWE-264 (23%) – Weaknesses and implementation mistakes in permissions, privileges, and access control;

CWE-200 (15%) – Information intentional or unintentional exposure to an actor that is not explicitly authorized to have access to that information;

CWE-20 (13%) – Improper input validation which may result in altered control flow, arbitrary code execution or illegal access to and control of resources;

CWE-399 (6%) – Improper management of system resources, e.g. memory allocation or reallocation;

CWE-189 (5%) – Numeric errors related to improper calculation or conversion of numbers;

CWE-362 (2%) – Concurrent code execution using shared resource with improper synchronization also knows as *Race Condition*;

CWE-310 (2%) – Cryptographic issues including missing encryption of sensitive data or key management errors;

CWE-94 (1%) – Improper control of code generation also known as *Code Injection* which often happens when software allows a user's input to contain code syntax.

CWE-416 (1%) – the use after free vulnerabilities, which result in referencing memory after it has been freed and can cause a program to crash, use unexpected values, or execute code.

Analysing both the quantity and CVSS severity scores of vulnerabilities of different type (see Fig. 8) we can conclude that the most critical ones are: CWE-119, CWE-264 and CWE-20. CWE-94, despite its small number, has the maximum severity on average (8.9).

Our analysis shows that CWE-119 vulnerabilities, also widely known as buffer overflow, still remain the most dominating and severe security flaws for all OSes. On the one hand, this can be explained by the fact that most of operating systems, written in C/C++, are prone to this type of weaknesses. On the other hand, it points to the fact that programmers neither really pay enough attention to such widely known problem that has been around for years nor follow best software development practices or make use numerous techniques proposed to cope with the buffer overflow issue.

As a result, vulnerabilities of the CWE-119 type (e.g. CVE-2016-7277, CVE-2016-4658 or CVE-2016-4598) often allow remote attackers to execute arbitrary code, read protected data or cause a denial of service.

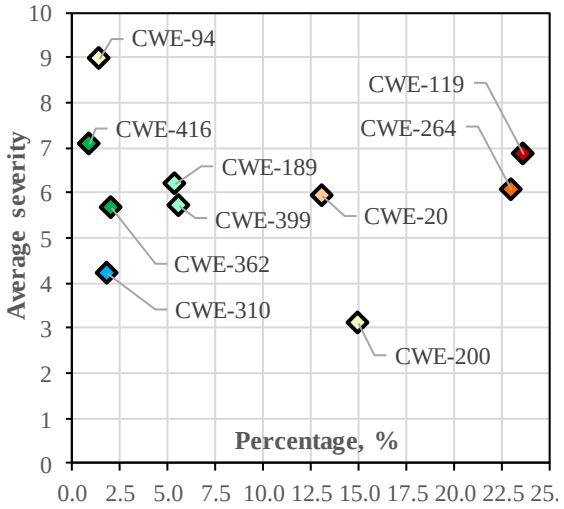


Fig. 8. The most numerous vulnerability types and their severity.

Distribution of different types of vulnerabilities for particular operating systems can be found in [13].

### *G. Common OS Vulnerabilities*

This section examines the vulnerabilities discovered in more than one operating systems by analysing CPE entries assigned to them. They are usually called common or shared [23, 24]. Common vulnerabilities and provide opportunity for compromising many or all of OSes at the same time and, being exploited, can cause a global epidemic of cyberattacks. They exist due to inheriting considerable parts of the OS code from its predecessor or reusing common components (system libraries, third party software components, OS kernels, etc.).

The common vulnerabilities are most often discovered in different releases of the same OS or in a family of related operating systems, e.g. BSD Unix (OpenBSD, FreeBSD, NetBSD) or Linux (Red Hat, CentOS, Novell, Ubuntu), etc.

For example, our analysis shows that 62 out of 63 (98%!) vulnerabilities reported by the NVD database in the most recent Apple MacOS 10.13 were also found in MacOS 10.8. The percentages of vulnerabilities shared between Microsoft Windows Server 2012 and its 2016th version is equal to 76% (123 vulnerabilities out of 165 ones found in Windows Server 2016 by the end of December 2017). It is remarkable that this number also includes 114 vulnerabilities (69%) that Windows Server 2016 shares with Windows Server 2008 and 23 vulnerabilities (14%) shared with Windows Vista. Moreover, at least six vulnerabilities in the SMB protocol, causing this year a massive WannaCry cyber attack, are traced to Windows Server 2003 and even to Windows XP.

The 6.x and 7.x (last updated on 01.08.2017) versions of the Red Hat Enterprise Linux, and 12.4 and 16.4 (released on 21.04.2016) versions of Ubuntu Server share up to 75% and 70% of common vulnerabilities correspondingly.

It is also worth noting that Oracle Solaris 11.3 in 2016 shared 29% of vulnerabilities with Oracle Solaris 10.0 and 24% with Oracle Linux 7 but none with the 11.0 version, analysed in the paper.

These results confirm that the developers of operating systems reuse significant pieces of code from the previous releases without really analysing their vulnerability or improving their security.

Sometimes hackers and security analysts discover vulnerabilities that are common for even different OS families. One of such vulnerabilities is CVE-2008-4609 found in October, 2008. It caused the denial-of-service attack for a variety of OSes and their versions, including Linux, BSD Unix, Microsoft Windows, Cisco IOS and possibly many others [41, 42]. The vulnerability manipulated the state of Transmission Control Protocol (TCP) connections exploiting an algorithmic error in protocol implementation in various operating systems. A remote attacker was able to cause connection queue exhaustion by flags manipulation in the TCP header of crafted network packets sent to a victim-computer.

Fig. 9 shows common vulnerabilities correlated between Linux and Unix operating systems during 2012–2017 (Windows did not share any vulnerabilities with the rest of studied OSes). Eighty-five of them were disclosed in all three Linux operating systems (Ubuntu, Novell and Red Hat) and ten were shared between Red Hat, Ubuntu and Solaris. Besides, there were six groups of vulnerabilities shared between different OS pairs: Ubuntu and Novell – 245, Red Hat and Ubuntu – 60, Novell and Red Hat – 36; Red Hat and MacOS – 245; Red Hat and Solaris – 4; Ubuntu and Solaris – 5.

These data emphasize the importance on analysing the vulnerabilities of diverse OSes.

The numbers in brackets correspond to those vulnerabilities observed in Linux kernels (the NVD database distinguishes between vulnerabilities observed in Linux-based operating systems and Linux-kernels). Thus, Fig. 9 clearly demonstrates that the largest number of common and group vulnerabilities shared between the Ubuntu, Novell and Red Hat OSes are those discovered in the Linux kernels (versions 3.2.x, 3.0.x and 2.6.32) used by them. In total, the percentage of common vulnerabilities shared between the three Linux OSes varies from 8% (for the 3-version system) to almost 45% (for the 2-version systems combining Ubuntu and Novell)!

It is also noteworthy that the two Unix-like operating systems, Solaris and MacOS, do not have common vulnerabilities at all while they share certain numbers of vulnerabilities with different Linux OSes.

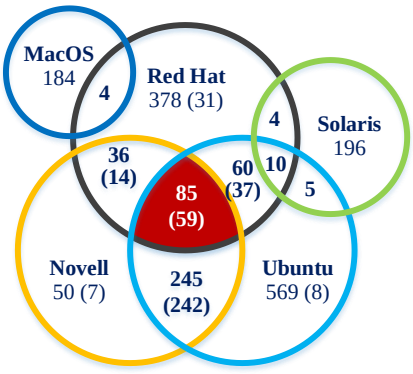


Fig. 9. Number of individual and common vulnerabilities shared by Linux (Ubuntu, Novell and Red Hat) and Unix (MacOS and Solaris) families of OSes.

The number of vulnerabilities shared by two or more OSes can be used as a measure of diversity between them [23]. Software diversity [18, 14, 21] has been used as a major fault and intrusion-tolerance mechanism to design safety-critical computer systems. Thus, choosing the most diverse OSes would allow to create the most secure and reliable multi-version system. Our empirical study demonstrates that vulnerability databases (the NVD database in particular) can help in determining the most diverse software products. Our analysis also shows that the results reported in [23] should be further verified as the authors may not have considered common and group vulnerabilities observed in Linux kernels.

## IV. Using OS Diversity to Improve System Security and Intrusion Tolerance

### A. OS Diversity and Intrusion Tolerance Architecture

Software vulnerabilities represent threats to dependability and, in particular, to security, that are additional to faults, errors and failures, traditionally dealt with by the dependability community [43, 44]. Design diversity is one of the most efficient methods for providing software fault-tolerance [14, 15] and improving dependability.

Often, researchers consider vulnerabilities as a special case of software faults activated by an attacker [44]. As a result, many studies focus on applying diversity to boost the intrusion tolerance of a system in the same way as software design diversity is used to ensure fault-tolerance.

In general, the diverse computer system consists of two or more replicas that run diverse software. The main assumption behind software diversity is that designs and implementations, developed independently (programmed by different teams, using diverse languages and development methodologies) will exhibit failure and vulnerability diversity.

Diversity, being a part of the intrusion tolerance mechanism, can improve system security, especially availability [23, 24, 45, 46]. However, the impact of software diversity on system confidentiality and integrity taking into account common vulnerabilities and the dynamic process of vulnerability discovery and patching is less understood.

There has been an increasing number of approaches and architectures proposed to build intrusion-tolerance systems. They employ different techniques to tolerate intrusions: adaptive redundancy and diversification principles [47, 48], asynchronous Byzantine agreement protocols [49, 50], replica “cleansing” [51], etc.

In our work we consider only one of many possible intrusion-tolerance architectures coping with vulnerabilities of operating systems. This architecture, shown in Fig. 10, comprises functionally redundant servers running diverse operating systems and a proxy/IDS that mediates client requests to all that servers and also verifies their behavior, as described in [45, 47, 52]. Intrusions are detected through the comparison of the server outputs before returning the result to the clients. This architecture suits well for tolerating intrusions in synchronous replicated server systems, e.g. intrusion-tolerant web servers.

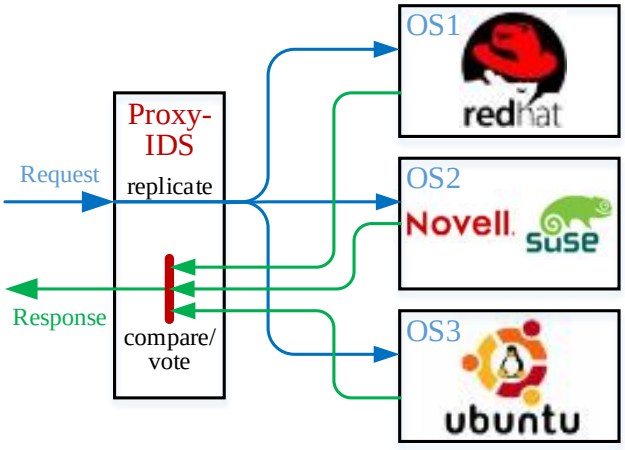


Fig. 10. Intrusion-tolerance architecture under study.

Operating systems of different families (e.g. Unix, Linux, Windows, MacOS) are more diverse, by nature, than those, belonging to the same OS family. However, using them for building a diverse intrusion-tolerance system usually causes various compatibility, portability and synchronization issues. This is why developers of diverse intrusion- and fault-tolerance systems often opt for using OSes of the same family [53, 54, 55]. In our study we examine a particular example of the diverse intrusion-tolerance architecture comprised of the three Linux-based OSes (Ubuntu, Novell and Red Hat), which common vulnerabilities were studied in Section III.G.

### *B. The Threat Model and Assumptions*

In the proposed intrusion-tolerance architecture (Fig. 10) all user requests and server responses synchronously pass through the proxy. The intrusion detection algorithm assumes that all noncompromised servers give the same answer to the same request [46, 47].

Thus, an intrusion is detected when the outputs are different due to an exploited vulnerability in one of diverse OSes. Majority voting is used then to identify a suspicious replica, isolate, cleanse/repair and reinsert it without interrupting a service. The general assumptions, which follow from the architecture description are:

--the system is synchronous; it does not need asynchronous Byzantine agreement protocols [50];

--data and states are replicated in all machines that simplifies system implementation; the system integrity and confidentiality can be further improved by applying threshold cryptography technique [56], however, it is out of the scope of this work ;

--an attacker cannot directly interact with a certain replica; all requests and responses go via the proxy;

--an attacker has only “one shot” at compromising the whole replicated system; a compromised replica, detected by IDS, is cleansed before an attacker will get a chance to compromise other replica(s) [46].

As follows from the above assumptions, if diverse OSes do not have common vulnerabilities (i.e. they are 100% diverse), a hacker would not be able to compromise all replicas at the same time (with the single malicious request). However, as the diverse replicas can share a certain number of common vulnerabilities, the least vulnerable diverse configuration is one with the minimal number of such vulnerabilities.

In the thread model we take into account the fact that a multi-version architecture can enlarge *the attack surface* (i.e. the total number of vulnerabilities that can be exploited) and, hence, can weaken system confidentiality and, sometimes, integrity [21]. Our threat model considers attack surfaces of a replicated diverse system for different types of attacks targeting availability, integrity and confidentiality in the following ways:

--the 3-replicated system preserves *availability* if at least one replica remains available (i.e. 1-out-of-3 replicas returns a response); thus, to make the system unavailable an attacker needs to target those vulnerabilities, common for all replicas, which impact availability (Fig. 11, a); attacking any other vulnerabilities would not make the entire diverse system unavailable;

--the 3-replicated system preserves *integrity* if 2-out-of-3 (the quorum) replicas return the correct response; thus, to compromise system integrity an attacker needs to target those vulnerabilities, common for any two replicas, which impacts *integrity* (Fig. 11, b);

--compromising any of diverse OSes would break the system confidentiality; thus, an attacker can target any vulnerability of any replica, which impact confidentiality (Fig. 11, c).

The attack surface of the 2-replicated diverse system has some differences depending on the system implementation (see Fig. 12):

--if a system is designed/configured to stop its operation once it detects data discrepancy (i.e. a fail-stop system [57]), an attack compromising integrity of 1-out-of-2 replica would make the whole system unavailable;

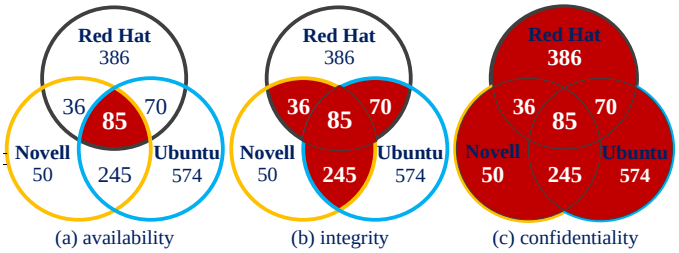


Fig. 11. Venn diagrams showing atack surface of the 3-version intrusion-tolerant system.

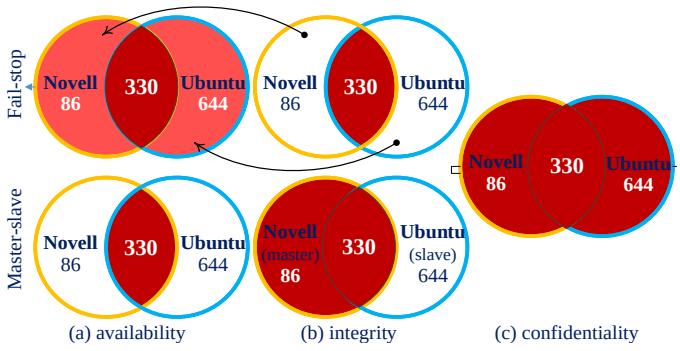


Fig. 12. Venn diagrams showing atack surface of the 2-version intrusion-tolerant system.

--if one of the OS versions is considered to be more trusted (a master replica), the system, when it detects inconsistency, will continue its operation using data provided by the more trusted master OS; the similar approach was used in the HACQIT project [52, 58]; in our study we assume that all 2-version architectures are configured as master-slave; OS having less number of discovered vulnerabilities is considered as a master replica.

Figs. 11 and 12 quantify attack surfaces of different security attributes using the static data (the overall number of individual and common vulnerabilities discovered in Linux-based OSes during 2012-2017) reported in Section III.G. As expected, the 3-version system architecture has the least number of common vulnerabilities (85). Among the 2-version systems the least vulnerable combination is Red Hat and Novell which has 121 of such vulnerabilities.

Ubuntu Server 12.04 and Novell Linux SUSE Enterprise Server 11 SP2 use similar versions of Linux core (3.2.x and 3.0.x) which explains their similarity in term of a number of common vulnerabilities (330).

### *C. Examining Static and Dynamic Impact of OS Diversity on Availability, Confidentiality and Integrity of the Intrusion-tolerance System*

In this section we quantitatively examine the vulnerability of several possible configurations of the intrusion-tolerance architecture, discussed above. As intrusion-tolerance servers are usually used to provide critical network services, in this section we consider only remotely exploitable vulnerabilities. Locally exploitable vulnerabilities identified based on their CVSS attack vector (see Section III.D for more details) are excluded from the study, as compared to Section III.A.

Table VI quantifies the network attack surface for individual OSes and various configurations of a diverse intrusion tolerant system taking into account vulnerability impact on different security properties. It clearly shows that developers of intrusion-tolerance systems deploying OS diversity have to trade-off between different security properties.

TABLE VI.
ATTACK SURFACES FOR DIFFERENT SECUROTY PROPERTIES IN VARIOUS DIVERSE CONFICURATIONS

| System architecture | OS | | | No of vulnerabilities | | |
|---|---|---|---|---|---|---|
| | Ubuntu | Novell | Red Hat | avail-ability | integrity | confiden-tiality |
| Single-version | * | | | 496 | 335 | 336 |
| | | * | | **145** | **91** | **99** |
| | | | * | 354 | 267 | 275 |
| Multi-version | * | * | | 90 | **91** | 376 |
| | * | | * | 77 | 267 | 560 |
| | | * | * | **46** | **91** | **329** |
| | * | * | * | **24** | 99 | 577 |

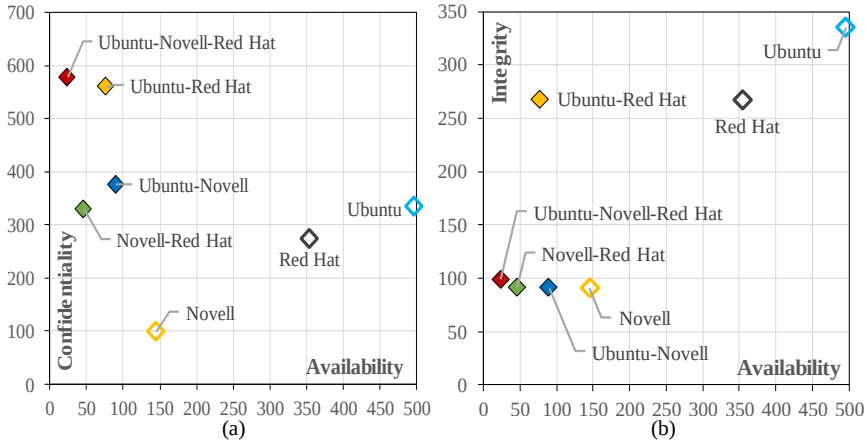


Fig. 13. Trade-offs between vulnerabilities impacting: a) availability and confidentiality, and b) availability and integrity

Fig. 13 demonstrates the interplay between a number of vulnerabilities affecting availability and confidentiality (Fig. 13.a), and availability and integrity (Fig. 13.b) for individual OSes and diverse configurations.

If one is ready to sacrifice confidentiality in favour of availability the 3-version architecture is the best choice. It provides also the best compromise between availability and integrity.

The pair Novell and Red Hat seems to be the best diverse configuration for maximising all the properties. It has the least number of vulnerabilities targeting integrity and confidentiality and also provides a good compromise with availability.

Among the individual OSes Novell has the least number of remotely exploitable vulnerabilities impacting availability, integrity and confidentiality. At the same time, Ubuntu should not be considered as a good choice in any scenario.

A more optimal decision regarding the best diverse configuration of the intrusion-tolerant system can be made dynamically by considering how many common vulnerabilities existed each day in a particular configuration (see Figs. 14-16).

Table VII summarises the statistics shown in Figs. 14-16 and provides arguments in favour and against each diverse configuration.

As we expected, the 3-version system significantly reduces a surface of network attacks targeting availability down to 1.08 vulnerabilities per day in average. It maintained the least number of forever-day vulnerabilities during the whole six-year period, during which 763 were days with no known vulnerabilities at all.

The combination of Novell and Red Hat is the best diverse configuration for a system which top priority is availability. On average, it maintains 2.04 vulnerabilities per day and ensures the same number of vulnerability-free days as the 3-version system.

The 3-version system still remains the best configuration for integrity-critical systems. It maintains 5 vulnerabilities per day on average. At the same time, the pair Ubuntu and Red Hat should not be considered as an appropriate option to build a diverse intrusion-tolerance system.

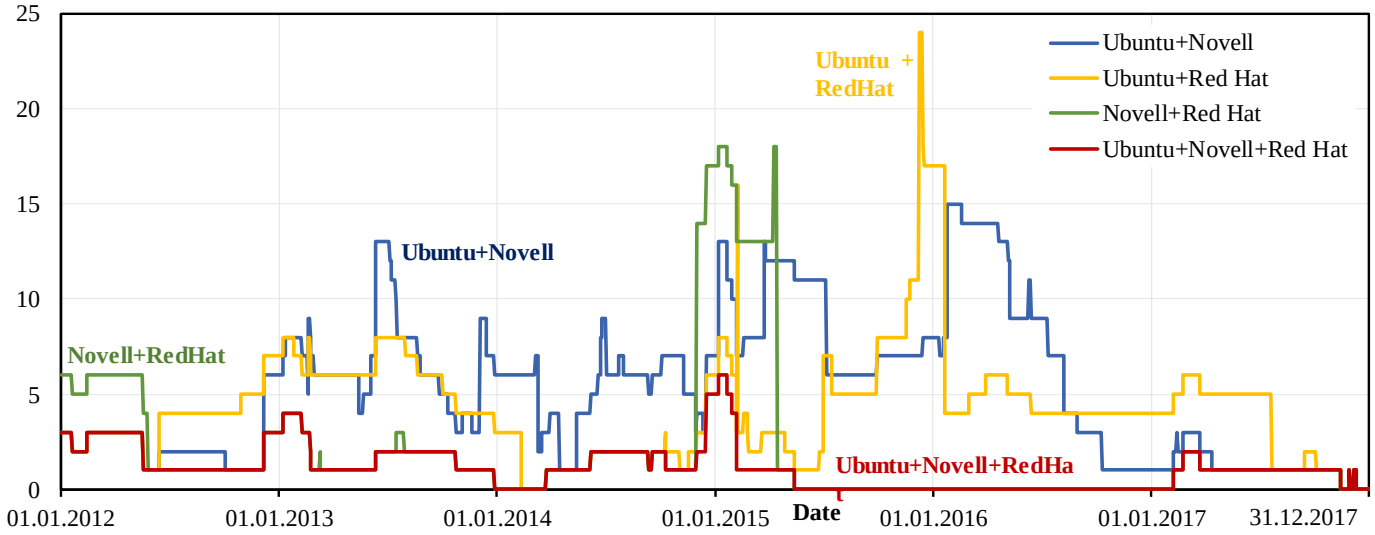


Fig. 14. Forever-day vulnerabilities in different configurations of a diverse intrusion tolerance system affecting availability.

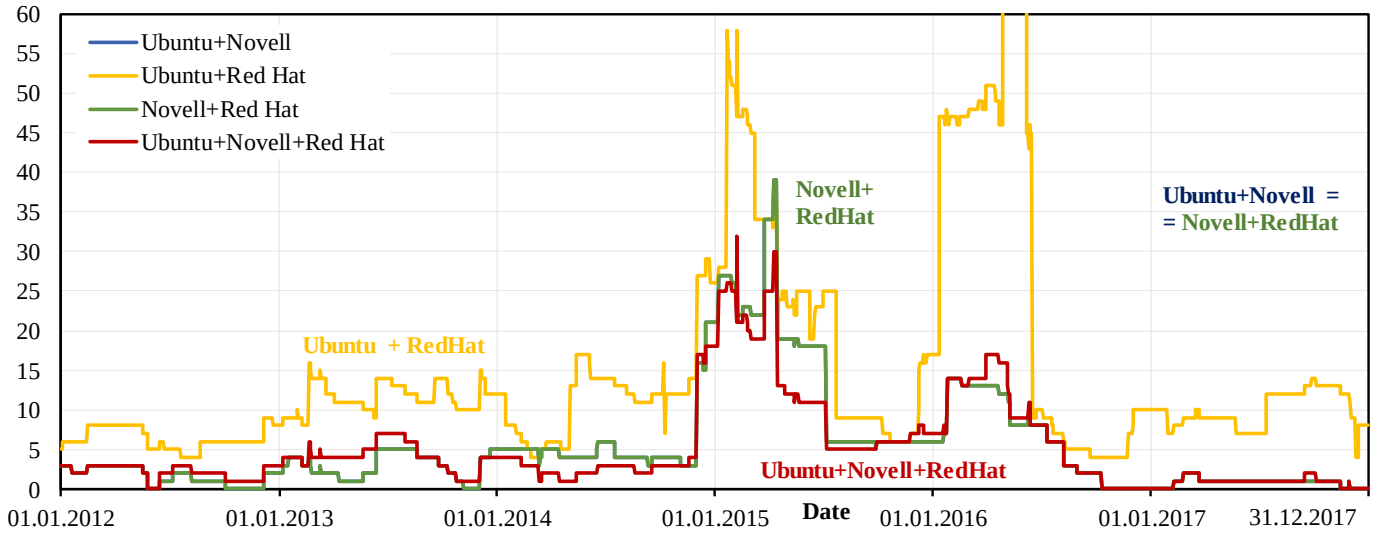


Fig. 15. Forever-day vulnerabilities in different configurations of a diverse intrusion tolerance system affecting integrity.

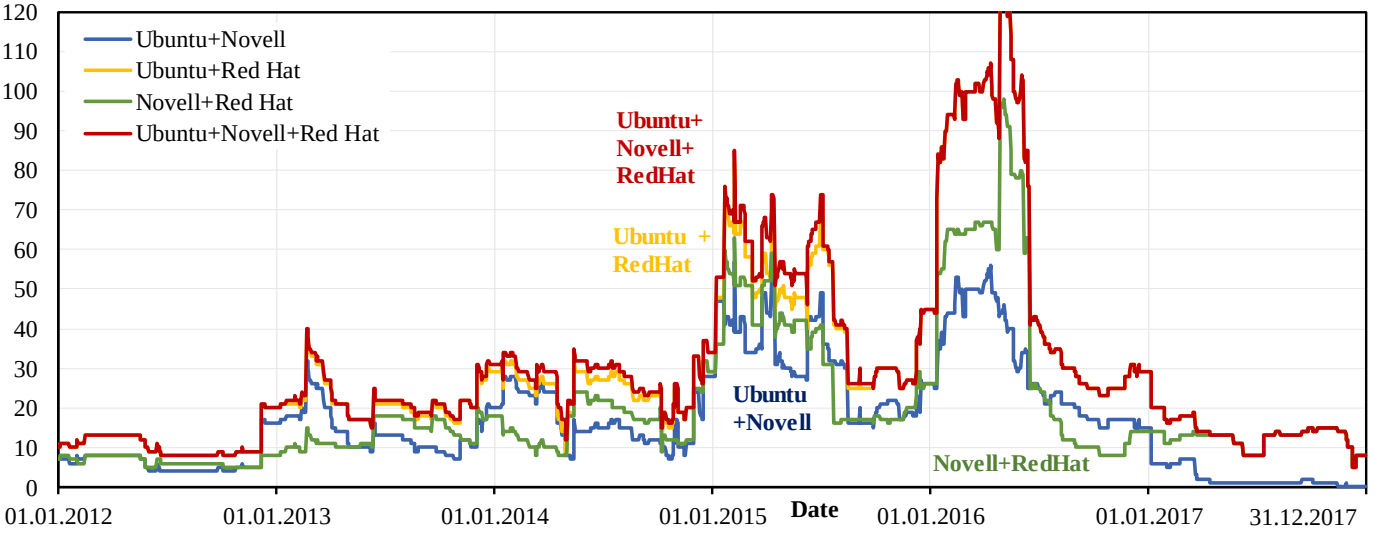


Fig. 16. Forever-day vulnerabilities in different configurations of a diverse intrusion tolerance system affecting consistency.

TABLE VII.
SUMMARY OF FOREVER-DAY VULNERABILITY STATISTICS (ATTACK SURFACE) FOR VARIOUS DIVERSE OS CONFIGURATIONS

| Operating System | | Diverse system configurations | | | |
|---|---|---|---|---|---|
| Ubuntu | | * | * | | * |
| Novell | | * | | * | * |
| Red Hat | | | * | * | * |
| Availability attack surface | | | | | |
| No of vulnerabilities per day | avg. | 5.24 | 4.26 | 2.04 | 1.08 |
| | min | 0 | 0 | 0 | 0 |
| | max | 15 | 24 | 18 | 6 |
| No of vulnerability free days | | 41 | 83 | 763 | 763 |
| No of days with the least number of forever-day vulnerabilities | | 521 | 566 | 1898 | 2192 |
| Integrity attack surface | | | | | |
| No of vulnerabilities per day | avg. | 5.31 | 14.82 | 5.31 | 5.01 |
| | min | 0 | 4 | 0 | 0 |
| | max | 39 | 83 | 39 | 32 |
| No of vulnerability free days | | 279 | 0 | 279 | 187 |
| No of days with the least number of forever-day vulnerabilities | | 1589 | 34 | 1589 | 1555 |
| Confidentiality attack surface | | | | | |
| No of vulnerabilities per day | avg. | 30.30 | 43.26 | 30.24 | 45.93 |
| | min | 9 | 9 | 0 | 9 |
| | max | 73 | 141 | 104 | 141 |
| No of vulnerability free days | | 47 | 0 | 0 | 0 |
| No of days with the least number of forever-day vulnerabilities | | 1587 | 0 | 849 | 0 |

Finally, a diverse system, for which the most important security property is confidentiality, would benefit from using either the combination of Novell and Red Hat, or Ubuntu and Novell. The first one had the least average number of forever-day vulnerabilities per day (30.24) affecting confidentiality, however the second one ensured 47 vulnerability-free days and maintained the minimal number of forever-day vulnerabilities during the longer period (1587 days versus 849 days).

The 3-version configuration is not recommended for use for the confidentiality-critical systems as it significantly enlarges an attack surface: up to 46 vulnerabilities per day in average.

## V. CONCLUSION AND LESSONS LEARNT

A significant growth of the total number of vulnerabilities discovered in modern OSes as well as the general tendency toward increasing their severity demonstrate the serious security challenges and risks that OS developers and users face.

It is very important to understand that the crucial parameters affecting system security are not only the total number of vulnerabilities disclosed in a particular software product and their severity but also, so called, *days-of-risk*, which show how fast software vendors issue patches fixing disclosed vulnerabilities, and a number of *forever-day vulnerabilities* defining the attack surface.

Our analysis shows that the average days-of-risk for the studied operating systems varies from 89 days for Ubuntu up to 130 days for Red Hat. Besides, it found that 28 forever-day vulnerabilities on average for the investigated OSes existed every day during 2012-2017 (a number of such vulnerabilities varies on average between 8 for Solaris and 48 for Ubuntu).

Thus, our work clearly supports our claim that decreasing days-of-risk and reducing a number of forever-day vulnerabilities is one of the main challenges in improving security of operating systems.

It is worrying that as our study shows, the rate with which OS developers issue security updates in general does not depend on vulnerability severity. Average days-of-gray-risk for the most critical vulnerabilities remains even 24% higher (!) than the one calculated for vulnerability of the lowest severity.

Another important finding is that developers reuse significant pieces of code from the previous releases (which is not surprising itself) without really analysing their vulnerability and improving their security. Moreover, buffer overflow vulnerabilities still remain the most dominant and severe security flaws for all OSes despite many techniques being proposed to cope with this type of vulnerabilities.

These our findings demonstrate the worrying shortcomings in the engineering practices and policies for developing security updates adopted by OS vendors, as well as, in the maintenance management processes they run.

Another specific aspect that the paper studies is the vulnerabilities that were discovered in more than one OSes. Such vulnerabilities, common for different operating systems and even different OS families, can lead to large-scale hacker attacks and virus epidemics.

This calls for application of specially-tailored intrusion-tolerance techniques. One of them is based on adopting software diversity. In the paper we quantitatively analyse how operating system diversity impacts attack surface taking into account individual and common vulnerabilities.

Unlike other studies, we investigate how diversity affects various security attributes: availability, integrity and confidentiality using historical statistics from the CVE and NVD vulnerability databases. We confirm that the more OS versions we use and the more diverse they are the more the system becomes tolerant to attacks targeting its availability. However, the diversity can undermine the integrity and confidentiality properties by enlarging system attack surface.

In particular, in our work we considered different possible configurations of 2- and 3-version intrusion-tolerance systems built by combining Linux-based OSes: Ubuntu, Novell and Red Hat.

Our practical findings based on real vulnerability statistics confirm that the 3-version architecture is the best choice to ensure high system availability and integrity. On average, it maintains only one forever-day vulnerability targeting system availability and five ones targeting data integrity. Correspondingly, it is 3.6 and 1.7 times less than the average results provided by individual OSes.

However, for the 3-version system the number of forever-day vulnerabilities targeting data confidentiality is 3.8 times larger. It is fair to note that even the best 2-version configuration (Ubuntu+Novell) enlarges the confidentiality attack surface by 2.1 times. These results show that OS diversity in certain scenarios can improve system intrusion-tolerance. Though, it is not a panacea for intrusions targeting integrity and, especially, confidentiality. This calls for developing more effective security mechanisms in addition to the traditional intrusion-tolerance solutions.

## VI. REFERENCES


[1] B. Clark, "Hackers take hospital offline, demand $3.6m ransom," [Online]. Available: http://thenextweb.com/insider/2016/02/15/hackers-take-hospital-offline-demand-3-6m-ransom/.

[2] C. Williams, "Passengers ride free on SF Muni subway after ransomware infects network, demands $73k," [Online]. Available: www.theregister.co.uk/2016/11/27/san_francisco_muni_ransomware/.

[3] National Audit Office, "Investigation: WannaCry cyber attack and the NHS," National Audit Office, UK, London, 2017.

[4] L. H. Newman, "The biggest cybersecurity disasters of 2017 so far," 01 July 2017. [Online]. Available: https://www.wired.com/story/2017-biggest-hacks-so-far/.

[5] MITRE Corporation, "Common Vulnerabilities and Exposures. Terminology," [Online]. Available: https://cve.mitre.org/about/terminology.html.

[6] Microsoft Inc., "Description of Software Update Services and Windows Server Update Services changes in content for 2016," 2016. [Online]. Available: https://support.microsoft.com/en-us/help/3215781/description-of-software-update-services-and-windows-server-update-services-changes-in-content-for-2016.

[7] J. Jones, "Days-of-risk in 2006: Linux, Mac OS X, Solaris and Windows," 2006. [Online]. Available: http://www.csoonline.com/article/2136935/data-protection/days-of-risk-in-2006---linux--mac-os-x--solaris-and-windows.html.

[8] P. Edmonds, "When It Comes to Protection from Vulnerabilities, Process Trumps "Many Eyes"," 2007. [Online]. Available: https://technet.microsoft.com/en-us/library/cc512608.aspx.

[9] A. Patrizio, "Report Says Windows Gets The Fastest Repairs," 2007. [Online]. Available: http://www.internetnews.com/security/article.php/3667201.

[10] J. Reavis, "Linux vs. Microsoft: Who Solves Security Problems Faster?," 2000. [Online]. Available: http://www.reavis.org/research/solve.shtml.

[11] J. Jones, "Basic Guide to Days of Risk," 2007. [Online]. Available: http://www.csoonline.com/article/2136934/data-protection/basic-guide-to-days-of-risk.html.

[12] D. Goodin, "Rise of "forever day" bugs in industrial systems threatens critical infrastructure," 2012. [Online]. Available: http://arstechnica.com/business/2012/04/rise-of-ics-forever-day-vulnerabiliiities-threaten-critical-infrastructure/.

[13] A. Gorbenko, A. Romanovsky, O. Tarasyuk and O. Biloborodov, "Experience Report: Study of Vulnerabilities of Enterprise Operating Systems," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE'2017)*, Toulouse, France, 2017.

[14] A. Avizienis, "The N-Version Approach to Fault-Tolerant Software," *IEEE Transactions on Software Engineering* , vol. 11, no. 12, pp. 1491-1501, 1985.

[15] A. Avizienis and J.-C. Laprie, "Dependable computing: From concepts to design diversity," *IEEE Proceedings*, vol. 74, no. 5, pp. 629-638, 1986.

[16] B. Littlewood, P. Popov and L. Strigini, "Design diversity: an update from research on reliability modelling," in *9th Safety-Critical Systems Symposium*, Bristol, UK, 2001.

[17] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault tolerance during execution," in *IEEE Computer Software and Applications Conference*, Kharagpur, 1977.

[18] B. Randell, "System Structure for Software Fault Tolerance," *IEEE Transactions on Software Engineering*, vol. 1, no. 2, pp. 221-232 , 1975.

[19] P. Popov, "Models of reliability of fault-tolerant software under cyber-attacks," in *IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, Toulouse, France, 2017.

[20] J. Dobson and B. Randell, "Building reliable secure computing systems out of unreliable insecure components," in *IEEE Conference on Security and Privacy*, Oakland, USA, 1986.

[21] B. Littlewood and L. Strigini, "Redundancy and diversity in security," in *9th European Symposium on Research Computer Security ( ESORICS'2004), LNCS 3193*, 2004.

[22] I. Gashi, A. Povyakalo, L. Strigini, M. Matschnig, T. Hinterstoisser and B. Fischer, "Diversity for Safety and Security in Embedded Systems," in *IEEE International Conference on Dependable Systems and Networks (DSN)*, Atlanta, USA, 2014.

[23] M. Garcia, A. Bessani, I. Gashi, N. Neves and R. Obelheiro, "OS Diversity for Intrusion Tolerance: Myth or Reality?," in *IEEE/IFIP 41st Int. Conf. on Dependable Systems & Networks (DSN'2011)*, 2011.

[24] M. Garcia, A. Bessani, I. Gashi, N. Neves and R. Obelheiro, "Analysis of Operating System Diversity for Intrusion Tolerance," *Software - Practice & Experience*, vol. 44, no. 6, pp. 735-770, 2014.

[25] S. Frei, M. May, U. Fiedler and B. Plattner, "Large-scale vulnerability analysis," in *SIGCOMM Workshop on Large-Scale Attack Defense*, 2006.

[26] M. Shahzad, M. Zubair Shafiq and A. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in *34th Int. Conf. on Software Engineering (ICSE '12)*, 2012.

[27] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *ACM Conference on Computer and Communications Security*, Raleigh, NC, 2012.

[28] B. Ladd, "The Race Between Security Professionals and Adversaries," 2017. [Online]. Available: https://www.recordedfuture.com/vulnerability-disclosure-delay/.

[29] A. Hahn and M. Govindarasu, "Cyber vulnerability disclosure policies for the smart grid," in *IEEE Power and Energy Society General Meeting*, San Diego, USA, 2012.

[30] M. Cheung, "Market Share Analysis: Server Operating Systems, Worldwide, 2015: Gartner report.," 2016. [Online]. Available: https://www.gartner.com/doc/3326217/market-share-analysis-server-operating.

[31] P. Tsai, "Server Virtualization and OS Trends," 2016. [Online]. Available: https://community.spiceworks.com/networking/articles/2462-server-virtualization-and-os-trends.

[32] W3Techs, "Usage of operating systems for websites," 2017. [Online]. Available: https://w3techs.com/technologies/report/operating_system.

[33] L. A. B. Sanguino and R. Uetz, "Software Vulnerability Analysis Using CPE and CVE," *Cryptography and Security*, vol. abs/1705.05347, 2017.

[34] M. Oiaga, "Recount: Windows Still Safest, Tops Mac OS X, Linux and Sun Solaris. But are statistics a true measure of security?," 2007. [Online]. Available: http://news.softpedia.com/news/Recount-Windows-Still-Safest-Tops-Mac-OS-X-Linux-and-Sun-Solaris-57433.shtml.

[35] J. Jones, "2006 Client OS Days of Risk," 2007. [Online]. Available: https://blogs.microsoft.com/microsoftsecure/2007/06/18/2006-client-os-days-of-risk/.

[36] H. Ghani, J. Luna and N. Suri, "Quantitative Assessment of Software Vulnerabilities Based on Economic-Driven Security Metrics," in *International Conference on Risks and Security of Internet and Systems (CRiSIS'2013)* , La Rochelle, France, 2013.

[37] A. Sajid, M. Ali Shah, M. Kamran, Q. Javaid and S. Zhang, "An Analysis on Host Vulnerability Evaluation of Modern Operating Systems," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 4, pp. 245-254, 2016.

[38] M. Kimura, "Software vulnerability: Definition, modelling, and practical evaluation for e-mail transfer software," *International Journal of Pressure Vessels and Piping*, vol. 83, pp. 256-261, 2006.

[39] H. Okamura, M. Tokuzane and T. Dohi, "Security Evaluation for Software System with Vulnerability Life Cycle and User Profiles (WDTS-RASD'2012)," in *Workshop on Dependable Transportation Systems/Recent Advances in Software Dependability*, Niigata, Japan, 2012.

[40] Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System, V3 Development Update," 2015. [Online]. Available: https://www.first.org/cvss.

[41] National Vulnerability Database, "Vulnerability Summary for CVE-2008-4609," 2008. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-4609.

[42] Cisco Systems, "TCP State Manipulation Denial of Service Vulnerabilities in Multiple Cisco Products," 2009. [Online]. Available:

https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20090908-tcp24.

[43] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11-33, 2004.

[44] P. Verissimo, N. Neves and M. Correia, "The Middleware Architecture of MAFTIA: A Blueprint," in *IEEE Information Survivability Workshop (ISW'2000)*, Boston, USA, 2000.

[45] F. Majorczyk, E. Totel and L. Me, "Experiments on COTS Diversity as an Intrusion Detection and Tolerance Mechanism," in *Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS'2007)*, Lisbon, Portugal, 2007.

[46] A. Valdes, M. Almgren, S. Cheung, D. Y., B. Dutertre, J. Levy, H. Saıdi, V. Stavridou and T. E. Uribe, "An Architecture for an Adaptive Intrusion-Tolerant Server," in *Security Protocols, LNCS 2845*, Berlin, Heidelberg, Springer-Verlag, 2002, pp. 158-178.

[47] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte and B. Dutertre, "An adaptive intrusion-tolerant server architecture," in *10th International Workshop on Security Protocols*, 2002.

[48] A. Saidane, V. Nicomette and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 45-58, 2009.

[49] F. Osorio, "Using Byzantine Agreement in the Design Of IPS Systems," in *IEEE International Performance, Computing, and Communications Conference*, New Orleans, USA, 2007.

[50] M. Correia, N. Neves and P. Verissimo, "BFT-TO: Intrusion Tolerance with Less Replicas," *The Computer Journal*, vol. 56, no. 6, pp. 693-715, 2013.

[51] Y. Huang, D. Arsenault and A. Sood, "Incorruptible system self-cleansing for intrusion tolerance," in *IEEE International Performance Computing and Communications Conference*, Phoenix, USA, 2006.

[52] E. Totel, F. Majorczyk and L. Me, "COTS Diversity Based Intrusion Detection and Application to Web Servers," in *Recent Advances in Intrusion Detection, LNCS 3858*, A. Valdes and D. Zamboni, Eds., Berlin, Heidelberg, Springer-Verlag, 2006, pp. 43-62.

[53] T. Distler, R. Kapitza and H. Reiser, "State transfer for hypervisor-based proactive recovery of heterogeneous replicated," in *'Sicherheit, Schutz und Zuverlässigkeit' Conference*, Berlin, 2010.

[54] M. Castro, R. Rodrigues and B. Liskov, " BASE: using abstraction to improve fault tolerance," *ACM Transactions on*, vol. 21, no. 3, pp. 236-269, 2003.

[55] C. Pu, A. Black, C. Cowan and J. Walpole, "A Specialization Toolkit to Increase the Diversity of Operating Systems," in *ICMAS Workshop on Immunity-Based Systems,*, Nara, Japan, 1996.

[56] A. Bessani, R. Mendes, T. Oliveira, N. Neves, M. Correia, M. Pasin and P. Verissimo, "SCFS: A Shared Cloud-backed File System," in *USENIX Annual Technical Conference*, Philadelphia, USA, 2014.

[57] R. Schlichting and F. Schneider, "Fail-stop processors: an approach to designing fault-tolerant computing systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 1, no. 3, pp. 222-238, 1983.

[58] J. Just and J. Reynolds, "HACQIT (Hierarchical Adaptive Control for QoS Intrusion Tolerance)," in *17th Annual Computer Security Applications Conference*, New Orleans, USA, 2001.

**Anatoliy Gorbenko**, received the M.Eng. degree in computer engineering in 2000 and the Ph.D. degree in computer science in 2005 from National Aerospace University, Kharkiv, Ukraine. He completed his D.Sc. habilitation and got a professorship with the Department of Computer Systems and Networks at National Aerospace University in 2012.

From 2014 to 2016, he was a dean of the Aircraft Radio-technical Faculty and leaded the Service-Oriented Systems Dependability research group. In 2017 Prof. Gorbenko joined the School of Computing, Creative Technologies & Engineering at Leeds Beckett University, UK. His expertise and research interests include dependability and performance of distributed systems, SOA and clouds; SW vulnerability and intrusion-tolerance.

**Alexander Romanovsky** received a M.Sc. degree in Applied Mathematics from Moscow State University and a PhD degree in Computer Science from St. Petersburg State Technical University. He was with this University from 1984 until 1996, doing research and teaching.

In 1993-94 he was a post-doctoral fellow with the Department of Computing Science, University of Newcastle upon Tyne, UK. Now he is a Professor in School of Computing, Newcastle University, UK and the Investigator of the PRiME programme and the STRATA platform EPSRC/UK grants. Prof. Romanovsky's main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system verification for safety, system structuring and verification of fault tolerance. He is a member of the editorial boards of Computer Journal, IEEE Transactions on Reliability, Journal of System Architecture and International Journal of Critical Computer-Based Systems.

**Olga Tarasyuk** received the B.S. degree in 2000, the M.S. degree in 2001 and the PhD degree in computing science in 2004 from National Aerospace University, Kharkiv, Ukraine.

Since 2005, she has been an Associate Professor with the Department of Computer Systems and Networks at National Aerospace University. Her main expertise is software quality and reliability, data analytics, development of dependable big-data and cloud computing solutions focusing on trade-offs between consistency, availability, performance and partition tolerance.

**Oleksandr Biloborodov** received the B.S. degree in 2011 and the M.S. degree in 2013 in computing science from National Aerospace University, Kharkiv, Ukraine.

Since 2014, he has been a senior software developer with Plarium LLC, Kharkiv, Ukraine. He is currently studying for the Ph.D. degree in computing science at National Aerospace University, Kharkiv, Ukraine. His main expertise is in quality of software and software engineering process, intrusion-tolerance and vulnerability analysis.